ASSIGNMENT 7

In this assignment, you are going to create a simple face cueing experiment. The goal of this assignment is to familiarize you with drawing images to canvas and with working with multiple canvases, which you are going to show on the screen sequentially with a small delay between each presentation.

This time and in the future, there might also be some questions you will have to answer in plain English (or Dutch). Put the answers to these questions in a *notepad* item which you place as the first item in your experiment

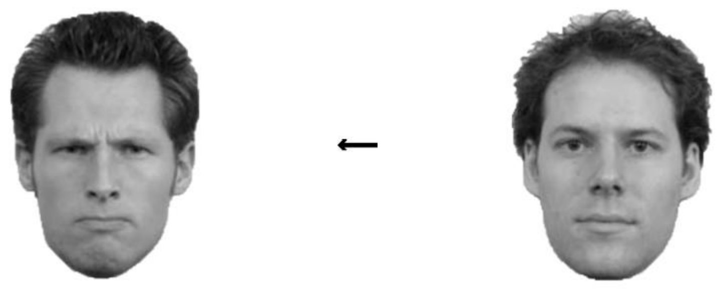
Once you are finished with this assignment, please save the file as

`Asgn_7_`**`<here your name>`**`.osexp`

(*for example* `Asgn_7_vanmoorselaar.osexp`)

before submitting it to canvas.vu.nl. Good luck!

Previous studies have shown that angry faces capture our attention (Schmidt, LJ, Belopolsky, AV, Theeuwes, J, 2012). That is, if we observe an angry face, we automatically focus our attention to it, even if we did not have any intention to do so at all in the first place. Our brain appears to be 'hard wired' to give angry faces priority of processing so to speak. The experiment you are going to create investigates if an angry face can serve as a cue for another stimulus. Participants will be shown two faces concurrently: one to the left and one to the right of fixation (i.e. the center of the screen). One of these faces looks angry, while the other has a neutral expression. Whether the angry face appears on the left or right will be randomly varied. Below you see an example of the *target display* (i.e. the final display that participants will have to respond to).



A short period after the faces have been presented, an arrow will appear in the center of the screen, replacing the fixation dot. This arrow is the stimulus that participants have to respond to: if the arrow points to the left, participants should press 'z' and if it points to the right they should press '/'. The hypothesis is that if the angry face captures attention, it likely *activates a response* for the side at which it appeared. If the arrow that appears a moment later thus points in the direction of the angry face, responses should be faster than when it points to the neutral face at the opposite side.

Procedurally, a trial in this experiment goes as follows:

- Present a display with a fixation dot (i.e. a fixation display) for 1000 ms
- Show the two faces for 100, 300, or 500 ms
- Reveal the arrow and wait for the participant's response

The independent variables we are going to manipulate in this experiment are:

• Arrow direction:	left *or* right
• Location of angry face:	left *or* right
• SOA (time between presentation of arrow and faces):	100,300 *or* 500 ms

Today we are going to make a full experiment, with a practice block and experimental blocks. The experiment will record responses from the participant and log them to a file.

1) Start by creating an empty *OpenSesame* experiment. You can choose to use the **Extended template** option in the New Experiment window. This saves you the steps of creating the basic structure of the experiment yourself, as this template already contains practice and experimental structures, which each contain the necessary block and trial loops. In addition, response and logger items have been placed at their commons positions. Easy as pie! Please set the Backend to Psycho.

2) Click on the item block_loop in the Overview window

a. Here you can set the variables by hand, but in the present case you have a full factorial design, so you can use the Full-factorial design button to fully cross the following *factors* and their *levels* (i.e. *variables* and their *values*). Once you click on the full factorial design button, then you can click on "Show example" to gain an idea about how to use the wizard.

i. arrow_direction - left, right
ii. angry_face_side - left, right
iii. SOA - 100, 300, 500

b. Manually, add a variable *correct_response* (with exactly this name! check this) and enter the value "z" in the rows which *arrow_direction* is *left* and a value "/" in the rows which *arrow_direction* is *right*.

c. Add a variable *cue_validity* and enter the value *valid* on the rows which *arrow_direction* and *angry_face_side* are both *left* or both *right*. On the rows in which *angry_face_side* and *arrow_direction* have a different value enter the value *invalid*. This variable *cue_validity* will thus indicate if the arrow pointed at the side at which the angry face was positioned. In this case, the angry face could be regarded as a valid cue for the direction of the arrow. Likewise, if the arrow pointed at the opposite side of the angry face, the angry face could be regarded as an *invalid* cue for the direction of the arrow.

3) Setting up the script and images

In the trial_sequence item, delete the sketchpad item and put an inline_script item in its place. Rename it from inline_script to *display_presentation*. Open this item.

a. Start with retrieving the values for the current cycle of the variables you put in block_loop with *var.* and store them in variables in the script. Do this for all variables except for *correct_response*. Correct_response will be handled automatically by OpenSesame.

Once you did this, try to run your program and print one of the variables using the *print* function (e.g. *print cue_validity*) in your inline script. Does it work?

- **b.** Download the three images of angry faces and three images of neutral faces from blackboard (all in png format). Place these 6 images file into the *File Pool*, so that you can use them in opensesame.
- **c.** Create two lists: *angry_faces* and *neutral_faces*. The first contains all the file names (as shown in the file pool) of the angry face images and the other those of the neutral face images.
- **d.** Randomly pick one item from the list containing angry faces and one item from the list containing neutral faces and store them in variables called *angry_face_image* and *neutral_face_image* respectively. For this you will have to import the module *random* at the top of your script and use its choice() function.
- **e.** Create three empty canvases: one to display the fixation display on, another one to show the two faces without the target arrow on and a final one to show the faces and the target arrow on. Store them in three separate variables. Give your variables appropriate names, like for example *fixation_canvas*.

4) Drawing the fixation and the face cue canvases

The next step will be to draw the actual canvases. What you need to draw of course depends on the values of the variables you retrieved from block_loop in 2a for the current cycle.

- **a.** On the fixation canvas, simply draw a fixation dot. That's all you need to do for this canvas. Note that there is a specific canvas function to draw the fixation dot (you can find it on the internet).
- **b.** On the face cue canvas:
  - i. Show *angry_face_image* on the side that corresponds to the value of *angry_face_side* and show *neutral_face_image* on the opposite side. If you can't remember how to display images on the canvas, look up the slides of lecture 4.
  - ii. Place the left image at ¼ of the screen width and the right image at ¾ of the screen width. Both images should be centered on the x-axis. Remember the slide about number division in Python 2!
  - iii. Scale up both images to be 1.5x their normal size.
  - iv. Check whether you can present the faces on your screen, by showing them in the run phase using the canvas.show() function. Does it work?

5) Drawing the target canvas

As you might have noticed, the target canvas is exactly the same as the face cue canvas, except for that the target arrow is now shown at the center of the screen instead of the fixation dot. You could of course repeat all the steps you took to create the face cue canvas, but you can also make a *copy* of the face cue canvas onto the empty canvas that you created to hold the target display. The latter is exactly what we are going to do.

- **a.** Look up the copy() function of Canvas in the OpenSesame documentation and figure out how it works.
- **b.** Make a copy of the face cue canvas on canvas you created to hold the target display.
- **c.** The finishing touches:
  - i. Draw a fixation dot in the center of the face cue canvas. Do you have an idea why we don't do it earlier?
  - ii. Draw the target arrow in the center of the target display canvas. The direction of the arrow should of course depend on the value you retrieved for *arrow_direction.* Make the arrow 40 pixels in length and set the arrow head size to 10.

6) Presenting the canvases on the screen

If all has gone well, you should now have three canvases.

1. One containing only a fixation dot in the center.
2. One containing a fixation dot in the center and a face on either side at a distance of ¼ screen width.
3. One containing an arrow in the center and a face on either side at a distance of ¼ screen width.

The next step is to draw all these canvases on the screen.
Switch to section of display_presentation in which you can put code for the run phase.

a. Show the canvas containing the fixation dot for 1000 ms.
b. Show the canvas containing the face cue for the duration of the value that you retrieved for *SOA*.
c. Show the canvas containing the target display until the participant has given a response.

7) The part in which we show the displays to the participant is complete, but now we have to process the response that participants give to the target display. Go to the keyboard_response item and

a. Make sure the Correct_response field is empty. Opensesame will determine the value of correct response by itself from the variable *correct_response* you have created in block_loop.
b. Set allowed responses to z;/ and leave Timeout on infinite

8) Change the settings of experimental_loop and block_loop in such a way that the experiment will have 16 blocks, each containing 24 trials (12 cycles repeated 2 times). Create some instructions and a break/experiment finished slides where appropriate.

All done: you now have a fully working experiment using an inline script.

BONUS QUESTION

9) We are now using the keyboard_response item to collect responses. As you may have figured, you can also collect responses using an inline script. Try to figure out how this works. Remove the keyboard_response item and collect responses via the inline script *display_presentation.*

10) This is one is a bit more tricky. Make sure that the feedback item still displays the feedback correctly. For today you only have to make sure that accuracy feedback is given (we focus on RT feedback in a later assignemt). Hint: Inspect the script of the feedback item to see which variables you need to update.